Introducción a Matlab

Primero pueden usar el prompt como una calculadora:
7+2 ans = 9 la respuesta se almacena en una variable llamada “ans”, que puede ser usada por ejemplo:
ans*ans ans = 81
Otro ejemplo: 1 + 3*2 - (1/2)*4
Se puede crear una función para su uso posterior:
sqr = @(x) x.^2;
z=@(x,y) sin(x)+cos(y);
Para ejecutar las funciones escriba, por ejemplo
a = sqr(5) a=25
z(0.2,0.3) ans=1.154
Se pueden generar **vectores** con igual espaciado de la siguiente forma:
1:5
0:2:10
0:.1:2*pi
Otra opción es usar linspace(start,stop,npoints):
x = linspace(0,1,10)

Matrices

```
a = [1 2 3;4 5 6;7 8 9]
a =
     1     2     3
     4     5     6
     7     8     9
```

También se pueden crear matrices a través de sub matrices

```
b = [a 10*a;-a [1 0 0;0 1 0;0 0 1]]
```

La función **repmat** se puede utilizar para replicar una matriz:

```
a = [1 2; 3 4]   repmat(a,2,3)
```

zeros: una matriz llena de ceros

```
zeros(2,3)
```

ones: una matriz llena de unos

```
ones(2,2)/2
```

rand: una matriz con elementos aleatorios uniformemente distribuidos

```
u = rand(1,5)
```

randn: una matriz con elementos aleatorios normalmente distribuidos

```
n = randn(5,5)
```

eye: matriz de identidad

```
eye(3)
```

Se puede trasponer una matriz **a** usando **a’**

Formas de graficar:

Ezplot: Función graficadora de fácil uso.

Sintaxis

ezplot(fun)) grafica la expresión *fun(x)* sobre el dominio default -2p < x < 2 p

ezplot(fun,[min,max]) grafica *fun(x)* sobre el domino: min < x < max

ezplot(fun2) grafica fun2(x,y) = 0 sobre el dominio default -2p < x < 2 p, -2 p < y < 2 p.

ezplot(fun2,[xmin,xmax,ymin,ymax]) grafica fun2(x,y) = 0 sobre xmin < x < xmax e ymin < y < ymax.

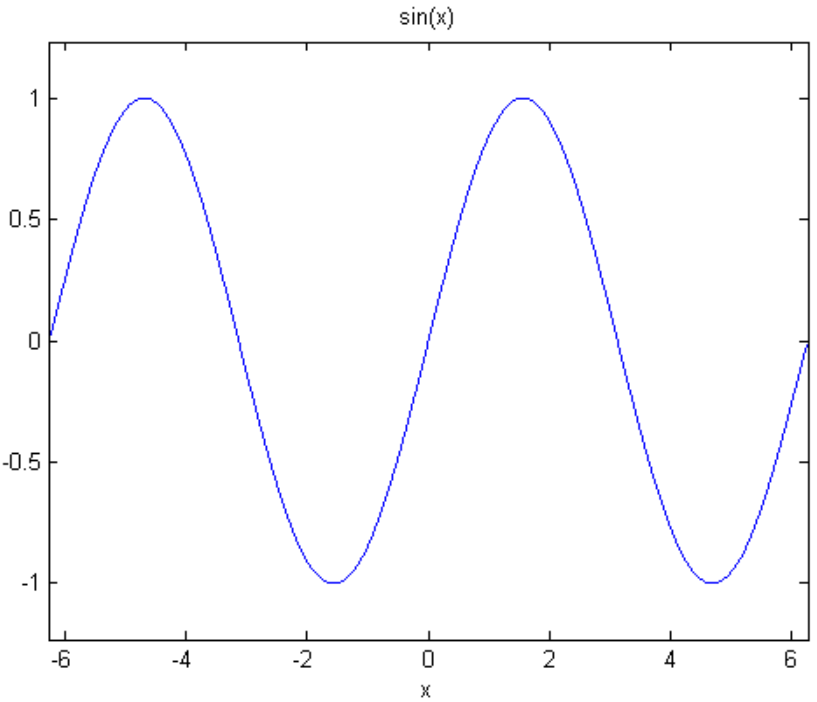
ezplot(fun2,[min,max])

ezplot(funx,funy) grafica la curva planar definida paramétricamente *funx(t)* y *funy(t)* sobre el dominio default 0 < t < 2p

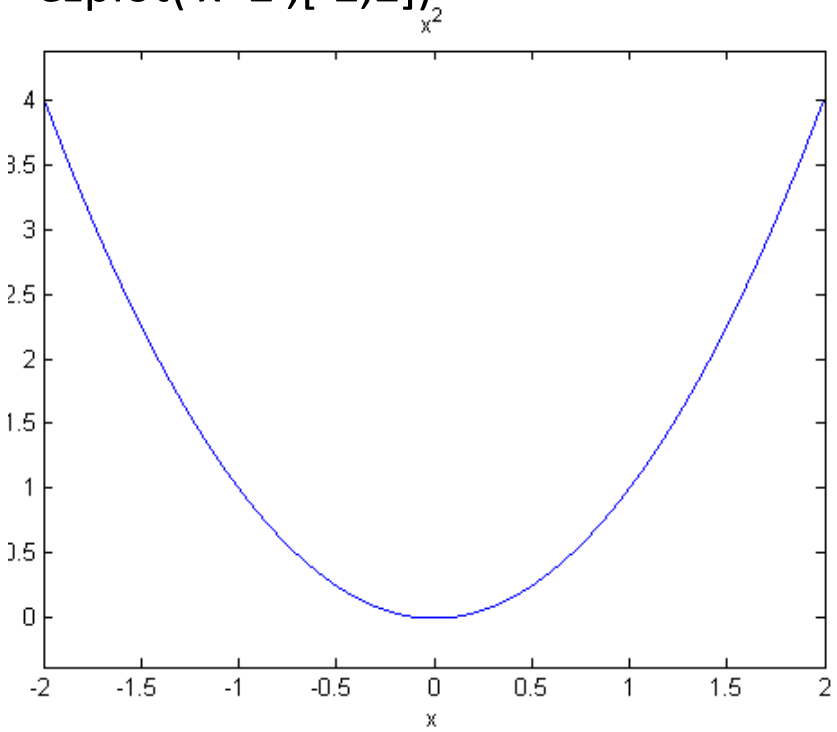
ezplot(funx,funy,[tmin,tmax]) grafica funx(t) y funy(t) sobre tmin < t < tmax

---

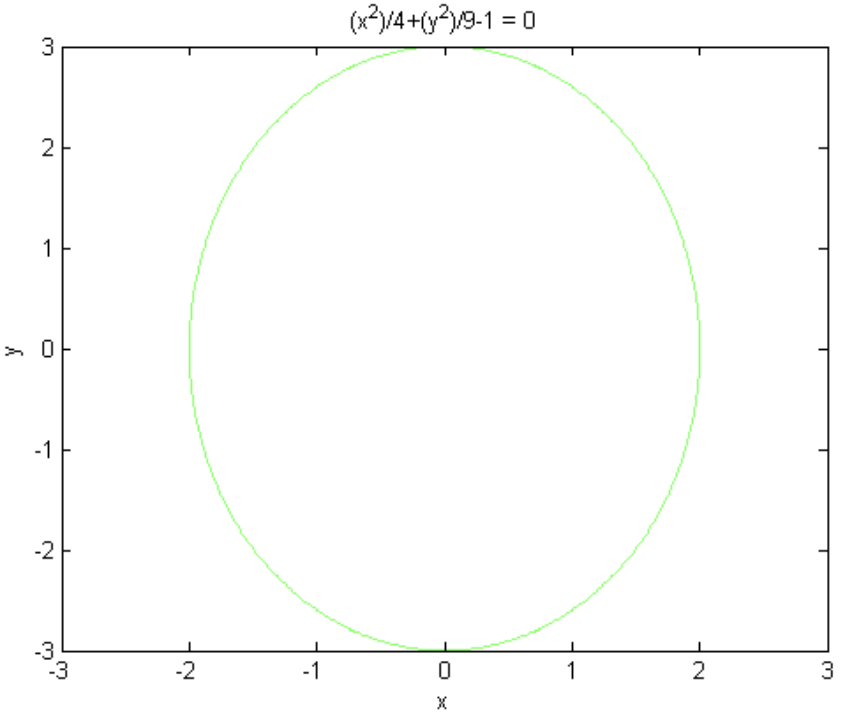
ezplot('sin(x)')



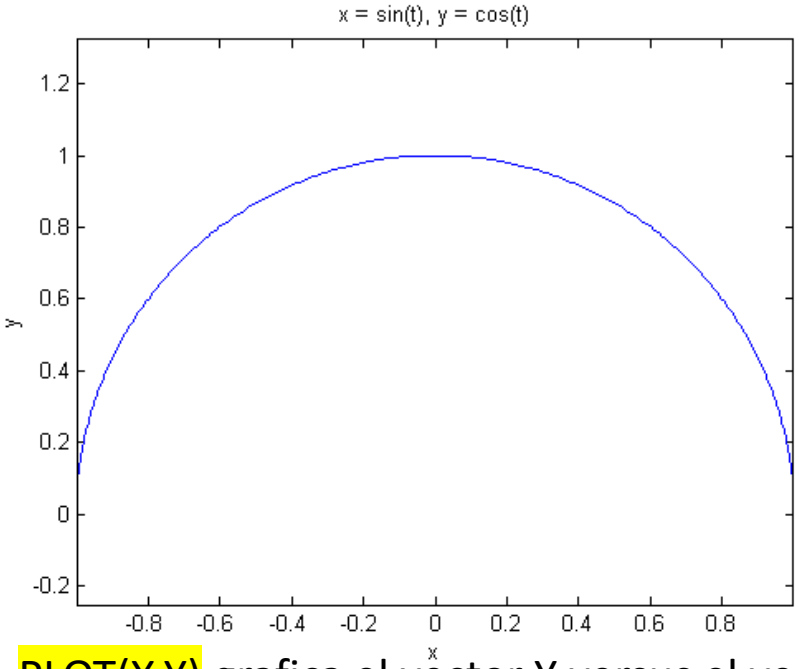
ezplot('x^2',[-2,2])



ezplot('(x^2)/4+(y^2)/9-1',[-3,3],[-3,3])



ezplot('sin(t)','cos(t)',[-1.5,1.5])

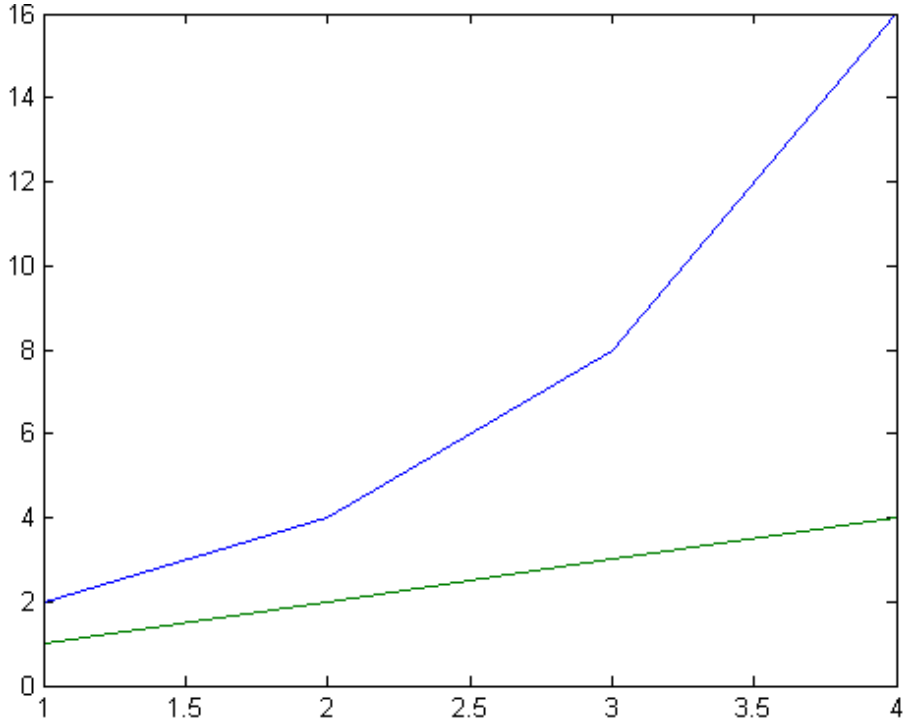


f1 = @(t) sin(t);f2 = @(t) cos(t);

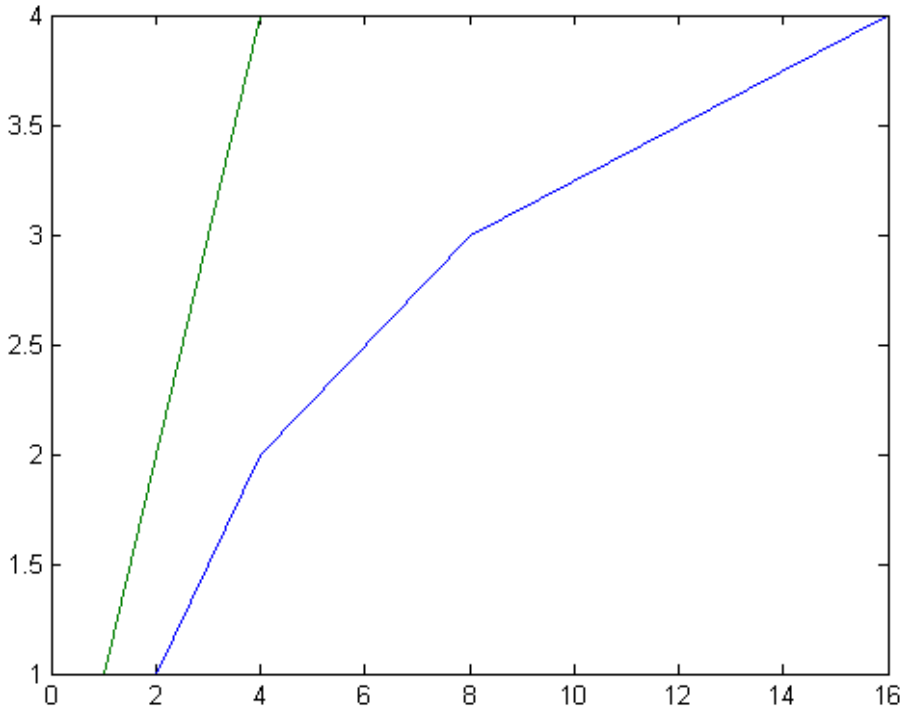
ezplot(f1,f2,[-1.5,1.5])

PLOT(X,Y) grafica el vector Y versus el vector X. si X o Y es una matriz, entonces el vector es graficado versus las filas o columnas de la matriz, lo que se alinea

```
x=[1 2 3 4];y=[2 4 8 16;1 2 3 4];
plot(x,y)
```

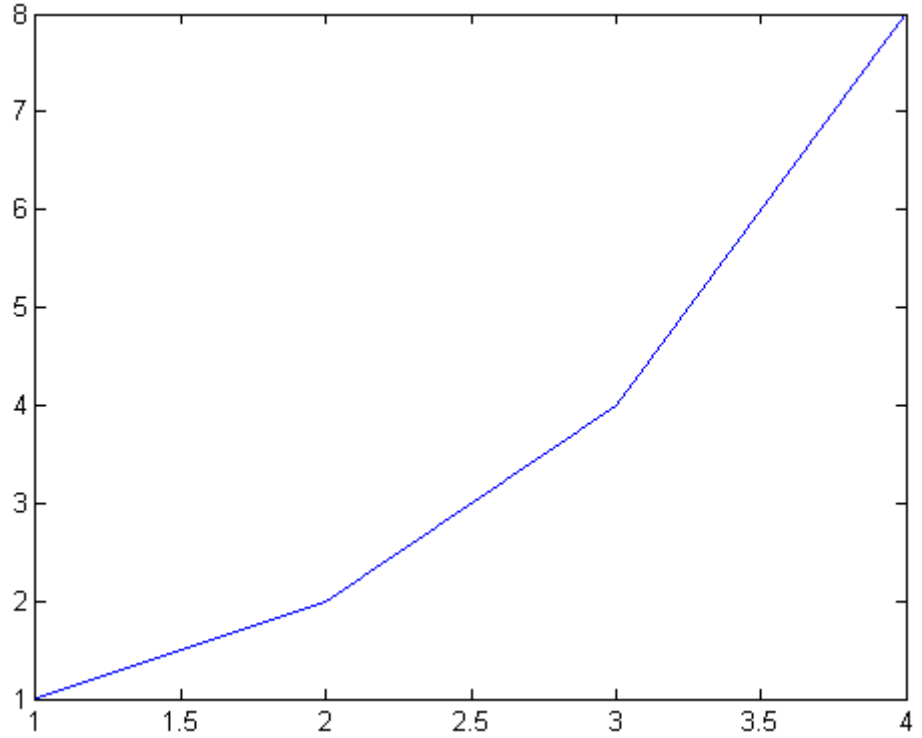


```
y=[1 2 3 4];x=[2 4 8 16;1 2 3 4];
plot(x,y)
```



PLOT(Y) grafica las columnas de Y versus su índice.

```
y=[1 2 4 8];
plot(y)
```



Si Y es complejo, PLOT(Y) es equivalente a PLOT(real(Y),imag(Y)).

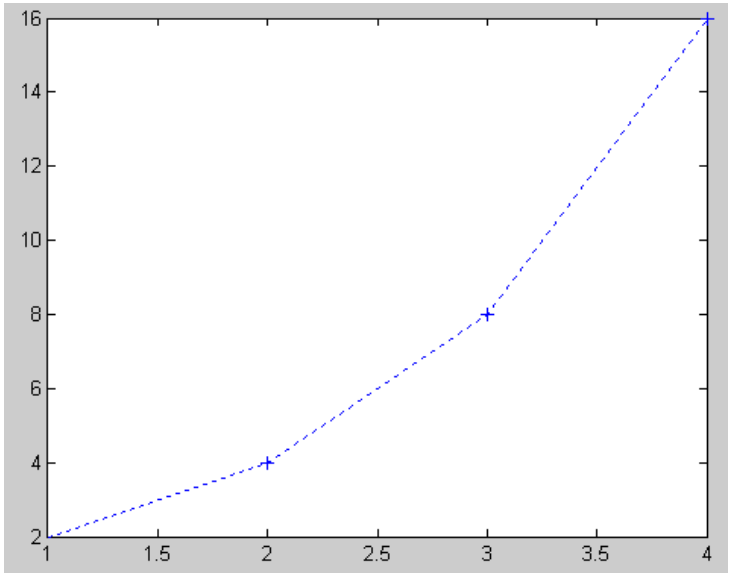
En todos los otros usos de PLOT, la parte imaginaria es ignorada.

```
x=-1:-1:-10;
y=log(x);
plot(x,y)
```

Se pueden obtener varios tipos de líneas, símbolos y colores con *PLOT(X,Y,S)* donde *S* es una cadena de caracteres hecha desde un elemento desde las siguientes 3 columnas:

b	blue	.	point	-	solid
g	green	o	circle	:	dotted
r	red	x	x-mark	-.	dashdot
c	cyan	+	plus	--	dashed
m	magenta	*	star	(none)	no line
y	yellow	s	square		
k	black	d	diamond		
w	white	v	triangle (down)		
		^	triangle (up)		
		<	triangle (left)		
		>	triangle (right)		

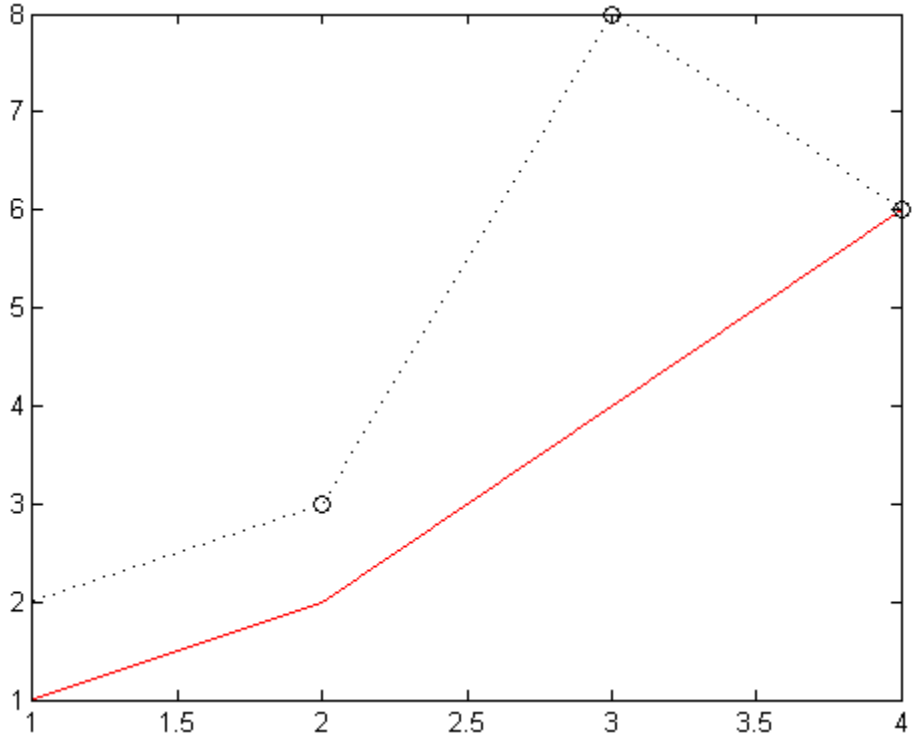
```
X=[1 2 3 4];Y=[2 4 8 16];
plot(X,Y,'b+:')
```



Grafica una línea de puntos (:) azul (b) con un signo más (+) en cada punto dato.

PLOT(X1,Y1,S1,X2,Y2,S2,X3,Y3,S3,...) combina los gráficos definidos por los triples *(X,Y,S)*, donde los X's e Y's son vectores o matrices y los S's son strings

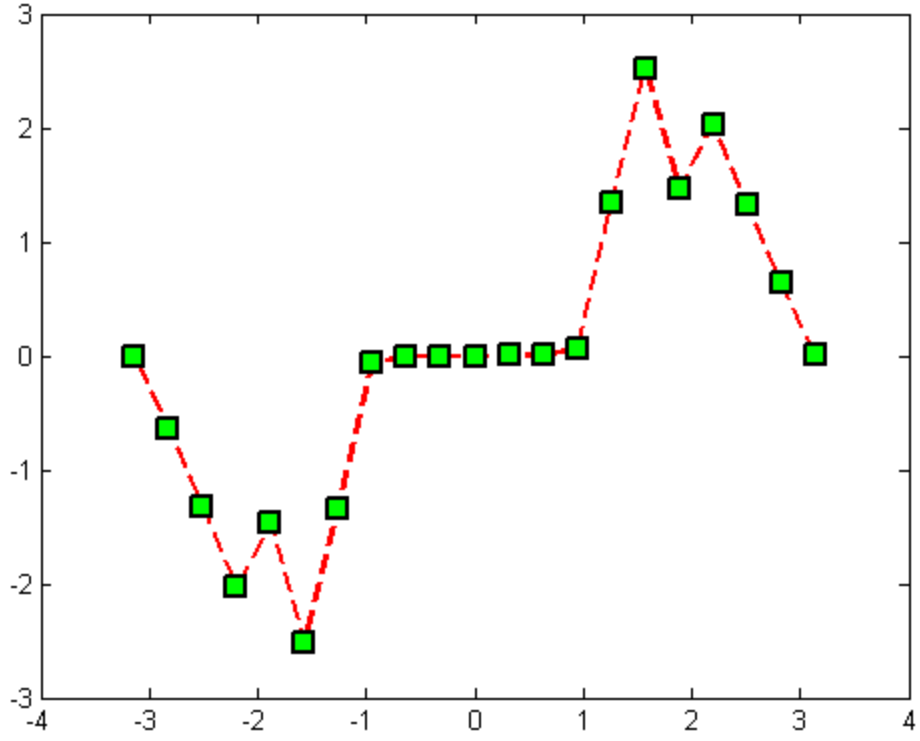
```
X=[1 2 3 4]; Y1=[1 2 4 6];Y2=[2 3 8 6];
plot(X,Y1,'r-',X,Y2,'k:o')
```



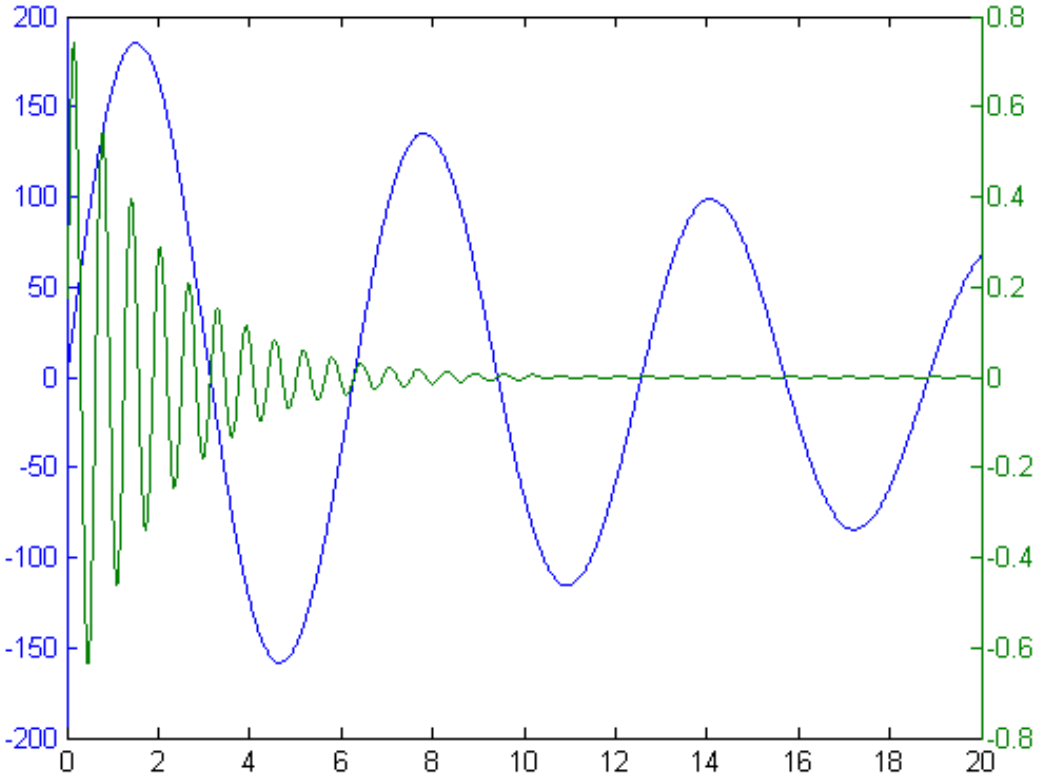
El comando PLOT, si no se especifica color, hace uso automático de colores especificados. Los default están listados en la tabla de arriba.

Si no se especifica tipo de marcador (marker type), PLOT no usa marcador. Si no se especifica un estilo de línea (line style), PLOT usa una línea sólida.

```
x = -pi:pi/10:pi;
    y = tan(sin(x)) - sin(tan(x));
    plot(x,y,'--rs','LineWidth',2,...
'MarkerEdgeColor','k',...
'MarkerFaceColor','g',...
                'MarkerSize',10)
```




```
x = 0:0.01:20;
y1 = 200*exp(-0.05*x).*sin(x);
y2 = 0.8*exp(-0.5*x).*sin(10*x);
[AX,H1,H2] = plotyy(x,y1,x,y2,'plot');
```

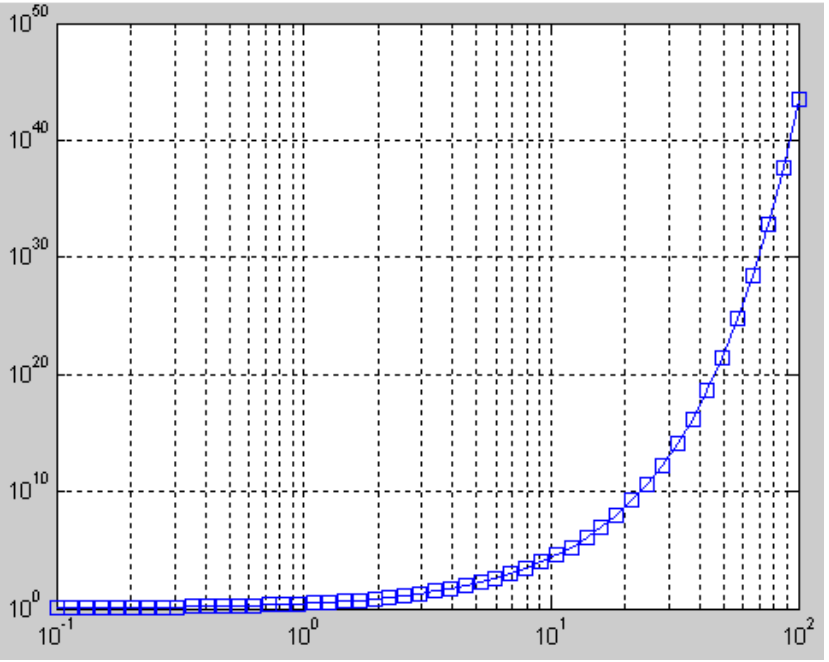


Loglog:

```
x = logspace(-1,2);

loglog(x,exp(x),'-s')

grid on
```

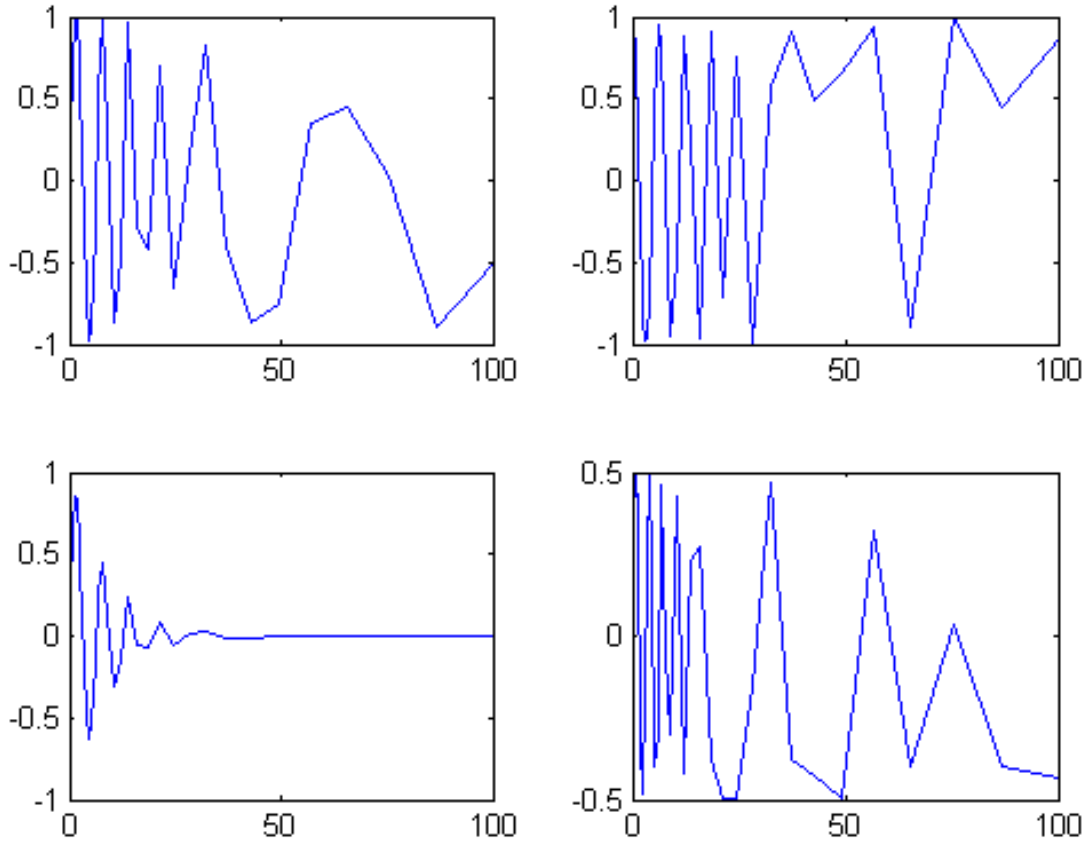


Existen además otras funciones orientadas a añadir títulos al gráfico, a cada uno de los ejes, a dibujar una cuadrícula auxiliar, a introducir texto, etc. Estas funciones son las siguientes:

title('título') añade un título al dibujo
xlabel('tal') añade una etiqueta al eje de abscisas. Con ***xlabel off*** desaparece
ylabel('cual') añade una etiqueta al eje de ordenadas. Con ***ylabel off*** desaparece
text(x,y,'texto') introduce 'texto' en el lugar especificado por las coordenadas **x** e **y**. Si **x** e **y**
son vectores, el texto se repite por cada par de elementos. Si **texto** es también un vector de cadenas de texto de la misma dimensión, cada elemento se escribe en las coordenadas correspondientes
gtext('texto') introduce **texto** con ayuda del ratón: el cursor cambia de forma y se espera un
clic para introducir el texto en esa posición
legend() define rótulos para las distintas líneas o ejes utilizados en la figura. Para más detalle, consultar el ***Help***
grid activa la inclusión de una cuadrícula en el dibujo. Con ***grid off*** desaparece la cuadrícula

Subplot: Una ventana gráfica se puede dividir en **m** particiones horizontales y **n** verticales, con objeto de representar múltiples gráficos en ella. Cada una de estas subventanas tiene sus propios ejes, aunque otras propiedades son comunes a toda la figura. La forma general de este comando es: **subplot(m,n,i)** donde **m** y **n** son el número de subdivisiones en filas y columnas, e **i** es la subdivisión que se convierte en activa. Las subdivisiones se numeran consecutivamente empezando por las de la primera fila, siguiendo por las de la segunda, etc.

```
y=sin(x); z=cos(x); w=exp(-x*.1).*y; v=y.*z;
subplot(2,2,1), plot(x,y)
subplot(2,2,2), plot(x,z)
subplot(2,2,3), plot(x,w)
subplot(2,2,4), plot(x,v)
```



Existen otras funciones gráficas bidimensionales orientadas a generar otro tipo de gráficos distintos de los que produce la función ***plot()*** y sus análogas. Algunas de estas funciones son las siguientes (para más información sobre cada una de ellas en particular, utilizar ***help nombre_función***):

bar() crea diagramas de barras

barh() diagramas de barras horizontales

bar3() diagramas de barras con aspecto 3-D

bar3h() diagramas de barras horizontales con aspecto 3-D

pie() gráficos con forma de “tarta”

pie3() gráficos con forma de “tarta” y aspecto 3-D

area() similar ***plot()***, pero rellenando en ordenadas de 0 a y

stairs() función análoga a ***bar()*** sin líneas internas

errorbar() representa sobre una gráfica –mediante barras– valores de errores

compass() dibuja los elementos de un vector complejo como un conjunto de vectores partiendo de un origen común

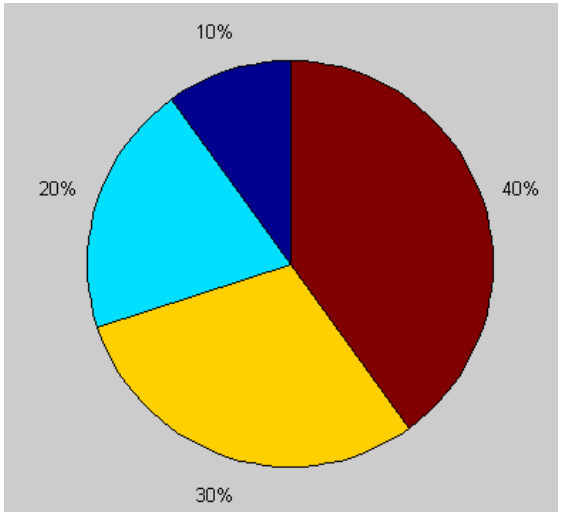
feather() dibuja los elementos de un vector complejo como un conjunto de vectores partiendo de orígenes uniformemente espaciados sobre el eje de abscisas

hist() dibuja histogramas de un vector

rose() histograma de ángulos (en radianes)

quiver() dibujo de campos vectoriales como conjunto de vectores

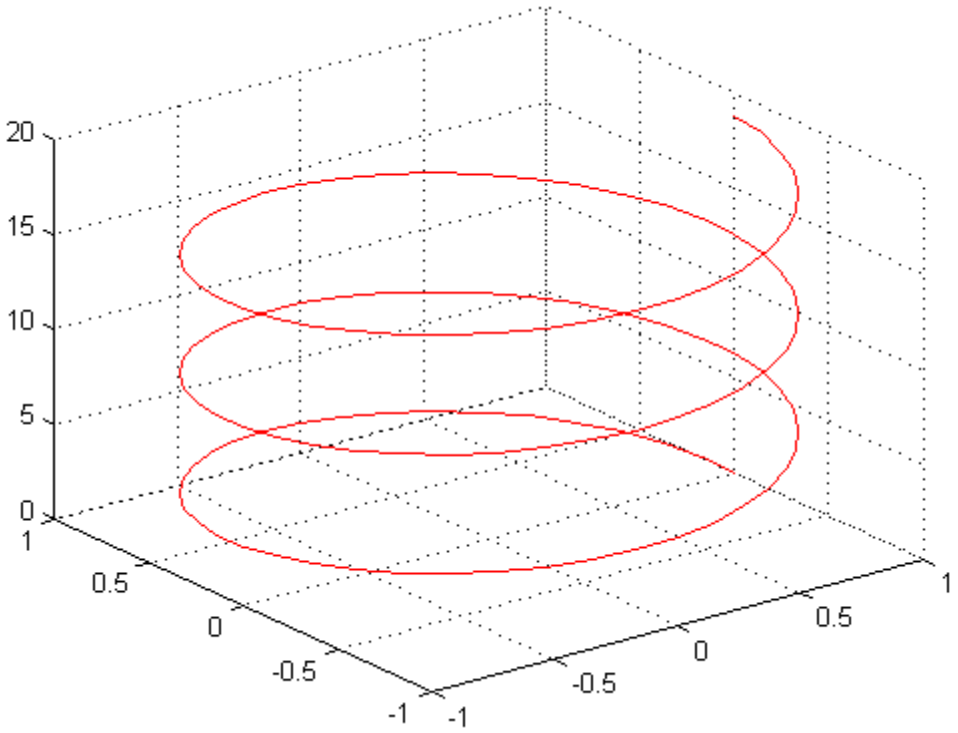
```
x=[1 2 3 4];
pie(x)
```



MATLAB tiene posibilidades de realizar varios tipos de gráficos 3D. Para darse una idea de ello, lo mejor es verlo en la pantalla cuanto antes, aunque haya que dejar las explicaciones detalladas para un poco más adelante.

La primera forma de gráfico 3D es la función ***plot3***, que es el análogo tridimensional de la función ***plot***. Esta función dibuja puntos cuyas coordenadas están contenidas en 3 vectores, bien uniéndolos mediante una línea continua (defecto), bien mediante ***markers***. Asegúrese de que no hay ninguna ventana gráfica abierta y ejecute el siguiente comando que dibuja una línea espiral en color rojo:

```
fi=[0:pi/20:6*pi];
plot3(cos(fi),sin(fi),fi,'r'), grid
```

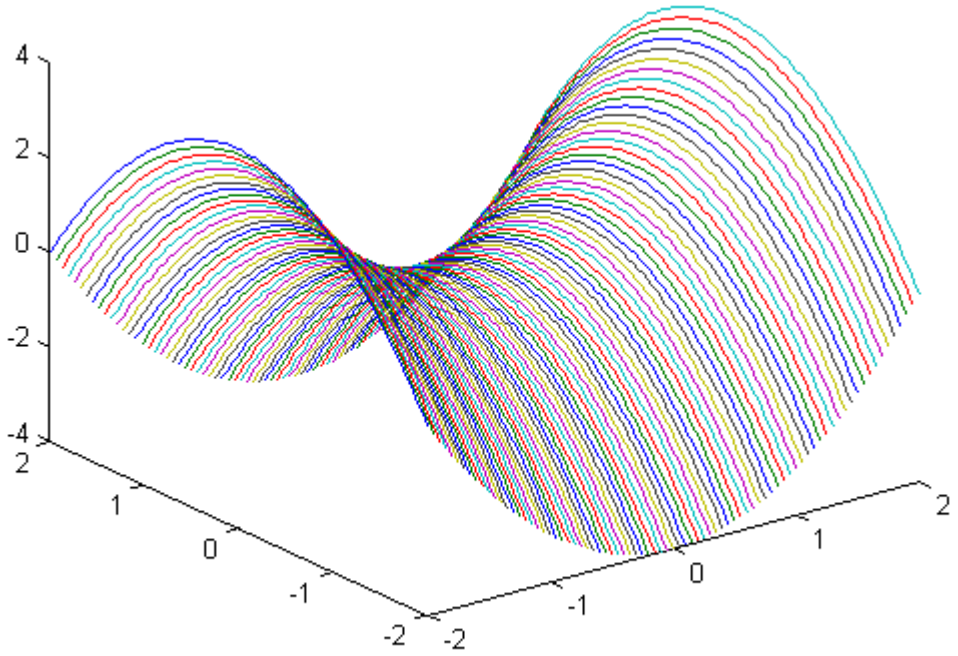


Función de dos variables

```
u=-2:0.05:2; v=u;
[U,V]=meshgrid(u,v);
z=@(x,y)x.^2-y.^2;
plot3(U,V,z(U,V))
```

Se puede usar:

```
surf(z(U,V))
```



Algunas funciones para análisis de datos

max máximum
min minimum
find find indices of nonzero elements
mean average or mean
median median
std standard deviation
sort sort in ascending order
sortrows sort rows in ascending order
sum sum of elements
prod product of elements
diff difference between elements

Algunas operaciones y relaciones lógicas

< menor que
<= menor o igual que
>mayor que
>= mayor o igual que
== igual a
~= distinto
& y
| o
~ no

SENTENCIA FOR

```
for variable = expresión
        <orden>
        <orden>        …
        <orden>
end
```

```
for x = 1:5
disp ('x toma el valor') % escribe por pantalla el
texto que se indica entre las comillas
disp (x) % escribe el valor de la variable x
end
```

```
x toma el valor
1
 x toma el valor
2
x toma el valor
3
x toma el valor
4
x toma el valor     5
```

SENTENCIA WHILE

Un bloque while ejecuta las órdenes mientras todos los elementos de la expresión sean verdaderos.

```
while <expresión>
<orden>
 <orden>
…
  <orden>
end
```

```
a=3;
while a < 5
disp ('a es menor que 5 ya que vale')
disp (a);
a = a + 1;
end
```

```
a es menor que 5 ya que vale
     3

a es menor que 5 ya que vale
     4
```

SENTENCIA IF

```
if <expresión>
 <órdenes evaluadas si la expresión es verdadera>
end
```

```
if <expresión>
 <órdenes evaluadas si la expresión es verdadera>
else
<órdenes evaluadas si la expresión es falsa>
end
```

```
if <expresión1>
<órdenes evaluadas si la expresión1 es verdadera>
elseif <expresión2>
<órdenes evaluadas si la expresión2 es verdadera>
…….
elseif        …
 else
 <órdenes evaluadas si ninguna otra expresión es verdadera>
end
```

```
b = 2;
        if b == 0 % ponemos == porque no es una asignación sino una expresión lógica
                disp ('b vale 0')
        elseif b == 1
                disp ('b vale 1')
        elseif b == 2
                disp ('b vale 2')
        elseif b == 3
                disp ('b vale 3')
        else
                disp ('b no vale ni 0 ni 1 ni 2 ni 3')
end
```

→b vale 2 % es lo que devuelve por pantalla

SENTENCIA BREAK: Si queremos que en un momento dado termine la ejecución de un bucle for o un bucle while usaremos break.